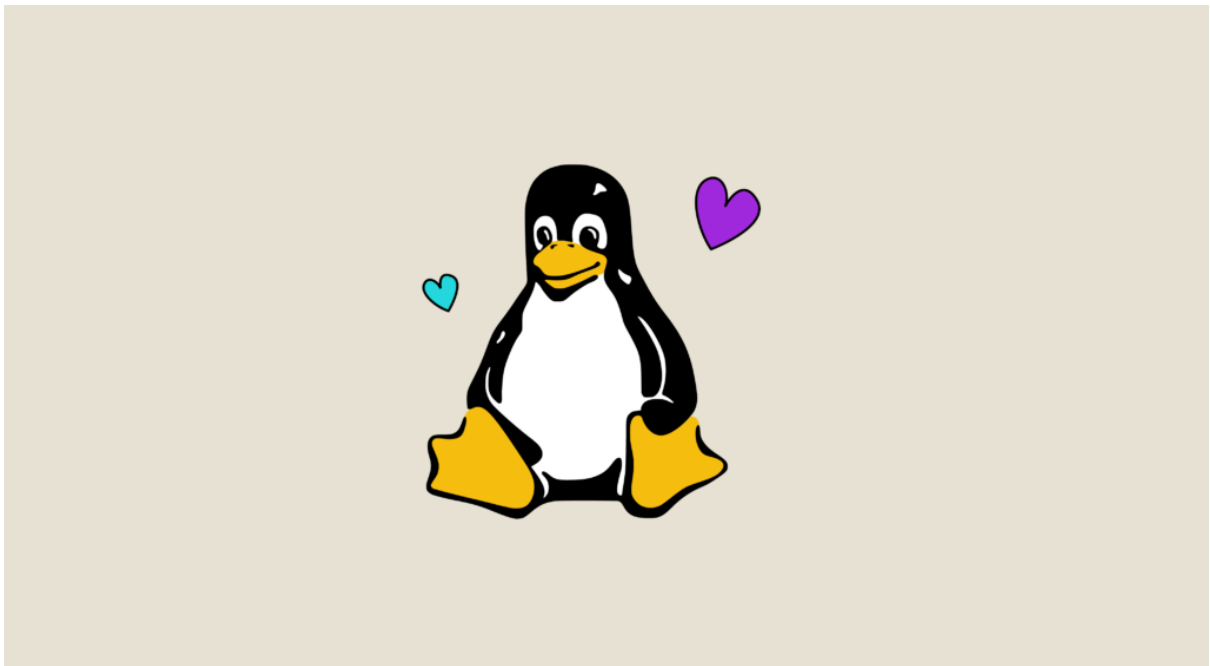


# Compte rendu TP de Linux embarqué

Valentin Lejars



J'adore Linux<3

# 1 Prise en main

## Séance 1

Avant toutes choses, la première étape pour utiliser la carte est de flasher la carte SD avec notre OS.

Pour cela j'ai utilisé Win32DiskImager, une fois cela fait j'ai pu insérer la carte SD, me connecter via la liaison série et reboot la carte pour observer la séquence d'allumage.

Ensuite j'exécute la commande `df -h` pour voir les partitions de ma mémoire:

```
root@OE10-Standard:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        3.0G  1.3G  1.5G  47% /
devtmpfs         375M    0  375M   0% /dev
tmpfs            376M    0  376M   0% /dev/shm
tmpfs            376M  9.8M  366M   3% /run
tmpfs            5.0M  4.0K  5.0M   1% /run/lock
tmpfs            376M    0  376M   0% /sys/fs/cgroup
tmpfs            76M    0   76M   0% /run/user/0
root@OE10-Standard:~#
```

On peut voir que notre OS n'a accès qu'à 3Go de mémoire.

J'exécute donc les commande pour élargir cette mémoire:

```
Command (m for help): Disk /dev/mmcblk0: 3.7 GiB, 3980394496 bytes, 7774208 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x55f3145b

Device      Boot   Start      End  Sectors  Size Id Type
/dev/mmcblk0p1  Boot   4096  1028095  1024000  500M b W95 FAT32
/dev/mmcblk0p2             1028096  7774207  6746112  3.2G 83 Linux
/dev/mmcblk0p3             2048      4095     2048    1M a2 unknown

Partition table entries are not in disk order.

Command (m for help): The partition table has been altered.
Calling ioctl() to re-read partition table.
Re-reading the partition table failed.: Device or resource busy

The kernel still uses the old table. The new table will be used at the next reboot or after you run partprobe(8) or kpartx(8).

Please reboot system and execute ./resize2fs_once.
root@OE10-Standard:~#
```

```
root@OE10-Standard:~# ./resize2fs_once
Starting resize2fs_once
resize2fs 1.42.13 (17-May-2015)
Filesystem at /dev/mmcblk0p2 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/mmcblk0p2 is now 843264 (4k) blocks long.

The filesystem has been enlarged upon.
root@OE10-Standard:~#
```

Pour la suite, on souhaite utiliser l'interface réseau de la carte VEEK.  
Après l'avoir branché je récupère son IP:

```
root@DE10-Standard:~# ipconfig
-bash: ipconfig: command not found
root@DE10-Standard:~# ifconfig
eth0      Link encap:Ethernet  HWaddr ca:e4:6d:9b:d8:45
          inet addr:192.168.88.70  Bcast:192.168.88.255  Mask:255.255.255.0
          inet6 addr: fe80::a21f:1eaf:df37:b869/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:71 errors:0 dropped:0 overruns:0 frame:0
          TX packets:26 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:7946 (7.9 KB)  TX bytes:2392 (2.3 KB)
          Interrupt:46

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:164 errors:0 dropped:0 overruns:0 frame:0
          TX packets:164 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:12228 (12.2 KB)  TX bytes:12228 (12.2 KB)

root@DE10-Standard:~#
```

192.168.88.70

Elle deviendra par la suite:

```
root@DE10-Standard:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 8e:a3:21:42:a2:7e brd ff:ff:ff:ff:ff:ff
    inet 192.168.88.67/24 brd 192.168.88.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::8ce3:21ff:fe42:a27e/64 scope link
        valid_lft forever preferred_lft forever
3: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1
    link/sit 0.0.0.0 brd 0.0.0.0

root@DE10-Standard:~#
```

192.68.88.67

```
# Logging
SyslogFacility AUTH
LogLevel INFO

# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
#AuthorizedKeysFile ~/.ssh/authorized_keys

# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts yes
# For this to work you will also need host keys in /etc/ssh_known_hosts
RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication no
# Uncomment if you don't trust ~/.ssh/known_hosts for RhostsRSAAuthentication
#IgnoreUserKnownHosts yes

# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords yes

root@OE10-Standard:~# :uq
-bash: :uq: command not found
root@OE10-Standard:~# ssh root@192.168.88.67
The authenticity of host '192.168.88.67 (192.168.88.67)' can't be established.
ECDSA key fingerprint is SHA256:YAVGTDiDJ5Pwbx1o4bkeYZtfVcVgKJIiTzuZVRnIJP4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.88.67' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.5.0-00198-g6b20a29 armv7l)

* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
Last login: Wed May 15 13:13:09 2024
root@OE10-Standard:~#
```

Après avoir édité le fichier /etc/network/interfaces je reboot la carte VEEK.

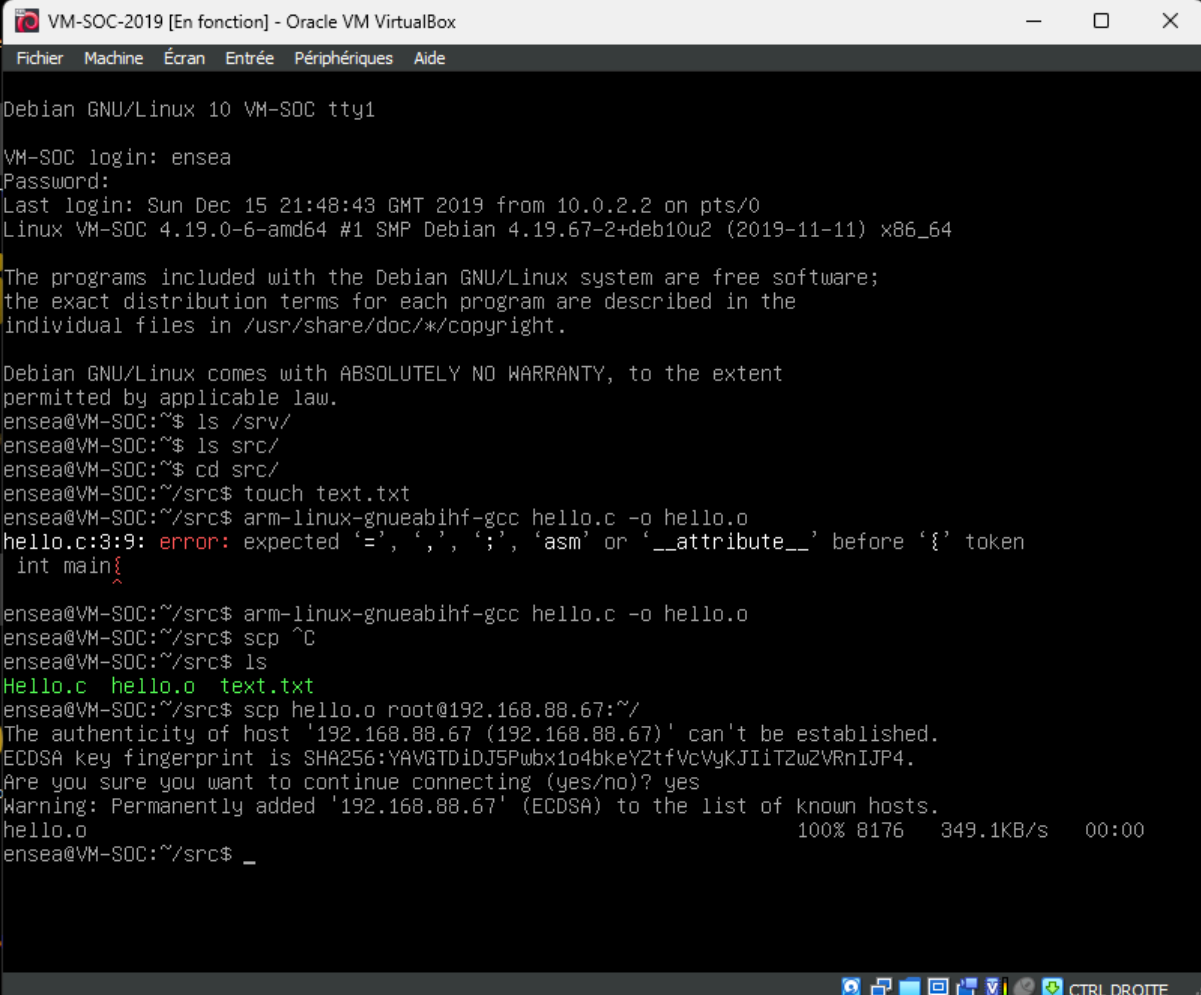
On veut maintenant utiliser une machine virtuelle pour pouvoir compiler des programmes en C avant de les envoyer sur la carte. J'installe donc Virtual Box.

J'écris mon premier code en C (j'ai personnellement codé sur NotePad++ mais c'est apparemment dépassé), il s'agit d'un simple hello World

```
#include <stdio.h>

int main(int argc, char *argv[]){
    printf("Hello World\n\r");
    return 0;
}
```

On compile sur la VM puis on envoie le programme sur la carte:



```
VM-SOC-2019 [En fonction] - Oracle VM VirtualBox
Fichier Machine Écran Entrée Périphériques Aide

Debian GNU/Linux 10 VM-SOC tty1

VM-SOC login: ensea
Password:
Last login: Sun Dec 15 21:48:43 GMT 2019 from 10.0.2.2 on pts/0
Linux VM-SOC 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
ensea@VM-SOC:~$ ls /srv/
ensea@VM-SOC:~$ ls src/
ensea@VM-SOC:~$ cd src/
ensea@VM-SOC:~/src$ touch text.txt
ensea@VM-SOC:~/src$ arm-linux-gnueabi-gcc hello.c -o hello.o
hello.c:3:9: error: expected '=', ',', ';', 'asm' or '__attribute__' before '{' token
int main{
^
ensea@VM-SOC:~/src$ arm-linux-gnueabi-gcc hello.c -o hello.o
ensea@VM-SOC:~/src$ scp ^C
ensea@VM-SOC:~/src$ ls
Hello.c hello.o text.txt
ensea@VM-SOC:~/src$ scp hello.o root@192.168.88.67:~/
The authenticity of host '192.168.88.67 (192.168.88.67)' can't be established.
ECDSA key fingerprint is SHA256:YAVGTDiDJ5Pwbx1o4bkeY2tfVcVyKJIiTzW2VRnIJP4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.88.67' (ECDSA) to the list of known hosts.
hello.o                                100% 8176   349.1KB/s   00:00
ensea@VM-SOC:~/src$ _
```

On peut ensuite exécuter le programme sur la carte:

```
root@OE10-Standard:~# ./hello.o  
Hello World  
root@OE10-Standard:~#
```

**INCROYABLE!!!** Ça fonctionne.

## Séance 2

Pour commencer cette nouvelle séance j'ai reconnecté directement la carte à Teraterm via le réseau. Puis j'ai demandé l'exécution du programme hello.c qui avait été réalisé la dernière fois.

```
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.5.0-00198-g6b20a29 armv7l)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
Last login: Wed May 15 14:12:09 2024 from 192.168.88.93
root@DE10-Standard:~# ./hello.o
Hello World
root@DE10-Standard:~# echo "1" > /sys/class/leds/fpga_led1/brightness
root@DE10-Standard:~#
```

La carte répond bien Hello World.

Ensuite j'ai utilisé une commande pour allumer une led sur la carte, celle-ci s'est bien allumée.

Pour le chenillard j'ai écrit un code en c:

```
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main(int argc, char *argv[]){
5      int i;
6      char str[] = "/sys/class/leds/fpga_led0/brightness";
7      FILE * file;
8
9      while(1)
10     {
11         for(i=0;i<10;i++)
12         {
13             sprintf(str, "/sys/class/leds/fpga_led%d/brightness", i);
14             file = fopen(str, "w");
15             fprintf(file, "1");
16             fclose(file);
17             usleep(100000);
18         }
19         for(i=0;i<10;i++)
20         {
21             sprintf(str, "/sys/class/leds/fpga_led%d/brightness", i);
22             file = fopen(str, "w");
23             fprintf(file, "0");
24             fclose(file);
25             usleep(100000);
26         }
27     }
28     return 0;
29 }
```

C'est un chenillard un peu custom qui allume toutes les LEDs une par une puis les éteint une par une. Il fonctionne, c'est joli.

## 2 Modules kernel

Pour commencer cette deuxième partie je fais à nouveau un chenillard, cette fois ci en utilisant le mmap.

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdint.h>
4  #include <sys/mman.h>
5  #include <fcntl.h>
6
7  int main(int argc, char *argv[]){
8      int i;
9
10     while(1)
11     {
12         for(i=0;i<10;i++){
13             {
14                 uint32_t * p;
15                 int fd = open("/dev/mem", O_RDWR);
16                 p = (uint32_t*)mmap(NULL, 4, PROT_WRITE|PROT_READ, MAP_SHARED,
17                     fd, 0xFF203000);
18                 *p = (1<<i);
19                 usleep(10000);
20             }
21             for(i=0;i<10;i++){
22                 {
23                     uint32_t * p;
24                     int fd = open("/dev/mem", O_RDWR);
25                     p = (uint32_t*)mmap(NULL, 4, PROT_WRITE|PROT_READ, MAP_SHARED,
26                         fd, 0xFF203000);
27                     *p = (0<<i);
28                     usleep(10000);
29                 }
30             }
31         }
32     }
33 }
```

Le code fonctionne mais se coupe au bout d'un moment en renvoyant "segmentation fault". Pour régler ce problème j'ai retiré les déclarations de \*p et fd de la boucle while pour les mettre au début du main.

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <stdint.h>
4  #include <sys/mman.h>
5  #include <fcntl.h>
6
7  int main(int argc, char *argv[]){
8      int i;
9      uint32_t * p;
10     int fd = open("/dev/mem", O_RDWR);
11
12     while(1)
13     {
14         for(i=0;i<10;i++){
15             {
16                 p = (uint32_t*)mmap(NULL, 4, PROT_WRITE|PROT_READ, MAP_SHARED,
17                     fd, 0xFF203000);
18                 *p = (1<<i);
19                 usleep(10000);
20             }
21             for(i=0;i<10;i++){
22                 {
23                     p = (uint32_t*)mmap(NULL, 4, PROT_WRITE|PROT_READ, MAP_SHARED,
24                         fd, 0xFF203000);
25                     *p = (0<<i);
26                     usleep(10000);
27                 }
28             }
29         }
30     }
31 }
```

Problème réglé.



On veut ensuite compiler nos propres noyaux dans la VM à partir des fichiers Makefile et hello.c sur moodle.

Pour cela il faut d'abord faire make puis on regarde que le hello.ko est bien là.

```
ensea@VM-SOC:~/src/module$ make
make -C /lib/modules/4.19.0-6-amd64/build M=/home/ensea/src/module modules
make[1]: Entering directory '/usr/src/linux-headers-4.19.0-6-amd64'
  CC [M]  /home/ensea/src/module/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
make[4]: Warning: File '/home/ensea/src/module/hello.mod.c' has modification time 0.075 s in the future
  CC      /home/ensea/src/module/hello.mod.o
  LD [M]  /home/ensea/src/module/hello.ko
make[4]: warning: Clock skew detected. Your build may be incomplete.
make[1]: Leaving directory '/usr/src/linux-headers-4.19.0-6-amd64'
ensea@VM-SOC:~/src/module$ ls
hello.c  hello.mod.c  hello.o  modules.order  timer_module.c
hello.ko  hello.mod.o  Makefile  Module.symvers
```

Puis on fait sudo insmod hello.ko et sudo rmmod hello:

```
ensea@VM-SOC:~/src/module$ sudo insmod hello.ko
ensea@VM-SOC:~/src/module$ sudo rmmod hello
[12517.447805] Bye bye...
```

Ces deux commandes servent respectivement à mettre hello.ko dans le noyau linux et à retirer le hello.

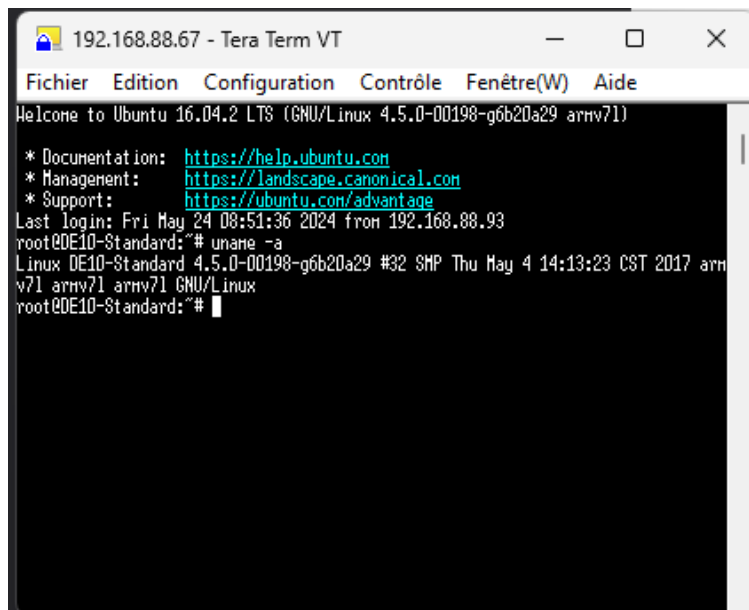
Pour voir le résultat on utilise la commande sudo dmesg:

```
[11735.091346] Hello world!
[11779.288179] Bye bye...
```

## Séance 3

2.3.0 Ces deux lignes servent à sélectionner la branche du git dont on sait qu'elle fonctionne avec notre noyau Terasic et à réduire la taille des identifiants de commit.

Puis je tape la commande uname -a pour voir la version de l'OS:



```
192.168.88.67 - Tera Term VT
Fichier  Edition  Configuration  Contrôle  Fenêtre(W)  Aide
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.5.0-00198-g6b20a29 armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
Last login: Fri May 24 08:51:36 2024 from 192.168.88.93
root@DE10-Standard:~# uname -a
Linux DE10-Standard 4.5.0-00198-g6b20a29 #32 SMP Thu May 4 14:13:23 CST 2017 armv7l armv7l armv7l GNU/Linux
root@DE10-Standard:~#
```

Après avoir installé ce qui est demandé en 2.3.1, on note le chemin:

```
Processing triggers for libc-bin (2.28-10) ...
ensea@VM-SOC:~$ whereis arm-linux-gnueabi-hf-gcc
arm-linux-gnueabi-hf-gcc: /usr/bin/arm-linux-gnueabi-hf-gcc
ensea@VM-SOC:~$
```

Mon super professeur m'a ensuite aidé à faire la partie 2.3.2 pour ne pas tout casser. Les lignes qui commencent par **export** servent à définir les variables environnement. Le tiret à la fin du chemin permet de mettre les noms des outils de compilation, ça devient -gcc.

Je récupère le chemin du noyau Kernel:

```
ensea@VM-SOC:~/linux-socfpga$ pwd
/home/ensea/linux-socfpga
ensea@VM-SOC:~/linux-socfpga$
```

Et je le remplace dans mon makefile:

```
1  obj-m:=hello.o
2  KERNEL_SOURCE=/home/ensea/linux-socfpga/
3  CFLAGS_MODULE=-fno-pic
4
5  all :
6      make -C $(KERNEL_SOURCE) M=$(PWD) modules
7  clean :
8      make -C $(KERNEL_SOURCE) M=$(PWD) clean
9  install :
10     make -C $(KERNEL_SOURCE) M=$(PWD) modules install
11
```

Je fais make puis je copie tout dans le répertoire src et une fois cela terminé je fais:

```
ensea@VM-SOC:~/module$ scp hello.ko root@192.168.88.67:
hello.ko                                100% 57KB 902.6KB/s 00:00
```

Avec dmesg on voit que ça s'exécute bien:

```
[ 19.463292] buffer0 virtual address 0x1b630000
[ 19.463320] buffer0 physical address 0x27400000
[ 5058.961143] Hello world!
root@OE10-Standard:~#
```

Je passe la partie du chenillard pour avoir le temps d'attaquer la 3ème partie.

### 3 Device tree

Après avoir modifié le fichier soc\_system.dts, avoir installé le device-tree-compiler et compilé le .dts en .dtb, je crée le dossier mntboot puis monte la partition.

Puis je renomme l'ancien soc\_system.dtb en .old.

```
root@OE10-Standard:~/mntboot# mv soc_system.dtb soc_system.old
root@OE10-Standard:~/mntboot# ls
System Volume Information  soc_system.old  soc_system.rbf  u-boot.scr  zImage
```

Ensuite j'envoie le nouveau .dtb vers la carte

```
ensea@VM-S0C:~/src$ scp soc_system.dtb root@192.168.88.67:
soc_system.dtb                                100% 27KB 1.2MB/s 00:00
```

Et, de retour sur la carte, je déplace mon nouveau fichier dans le bon répertoire:

```
root@OE10-Standard:~# cp /root/soc_system.dtb /root/mntboot/
root@OE10-Standard:~# cd mntboot/
root@OE10-Standard:~/mntboot# ls
System Volume Information  soc_system.old  u-boot.scr
soc_system.dtb            soc_system.rbf  zImage
```

Après le redémarrage de la carte je vais consulter le fichier soc@0:

```
root@OE10-Standard:/proc# cd device-tree/
root@OE10-Standard:/proc/device-tree# ls
#address-cells  aliases  clocks  cpus  model  soc@0
#size-cells     chosen  compatible  memory  name
root@OE10-Standard:/proc/device-tree# cd soc@0
root@OE10-Standard:/proc/device-tree/soc@0# ls
#address-cells  flash@0xff705000  i2s@0  spi@0xffff01000
#size-cells     flash@0xff900000  intc@0xfffed000  sysmgr@0xffd08000
L2-cache@0xffffef000  fpga@bridge@0  leds  timer@0xffc08000
bridge@0xc0000000  fpga@bridge@01  name  timer@0xffc09000
bus-frequency  fpga@bridge@02  pmu@0  timer@0xffd00000
can@0xffc00000  fpga@bridge@03  ranges  timer@0xffd01000
can@0xffc01000  fpga@bridge@04  r13regs@0xff800000  timer@0xffd02000
clkgr@0xffd04000  gpio@0xff708000  rstgr@0xffd05000  timer@0xffd03000
compatible  gpio@0xff709000  scu@0xfffec000  timer@0xfffec600
device_type  gpio@0xff70a000  sdr-ctl@0xffc25000  usb@0xffb00000
dma@0xffe01000  i2c@0xffc04000  serial@0xffc02000  usb@0xffb40000
ethernet@0xff700000  i2c@0xffc05000  serial@0xffc03000  usbphy@0
ethernet@0xff702000  i2c@0xffc06000  sound  vcc3p3-regulator
flash@0xff704000  i2c@0xffc07000  spi@0xffff00000
```